aurum
Revista Multidisciplinar

ISSN: 3085-7783

INTELIGÊNCIA ARTIFICIAL GENERATIVA NA TRANSFORMAÇÃO DO DESENVOLVIMENTO DE SOFTWARE: IMPACTOS, DESAFIOS E PERSPECTIVAS FUTURAS

GENERATIVE ARTIFICIAL INTELLIGENCE IN THE TRANSFORMATION OF SOFTWARE DEVELOPMENT: IMPACTS, CHALLENGES, AND FUTURE PERSPECTIVES

https://doi.org/10.63330/armv1n5-015

Submetido em: 19/07/2025 e Publicado em: 24/07/2025

José Henrique Salles Pinheiro
Nível Superior
UniCV
E-mail: aprovados_concursos@yahoo.com

RESUMO

A inteligência artificial generativa representa uma das mais significativas transformações tecnológicas do século XXI, especialmente no campo do desenvolvimento de software. Este trabalho apresenta uma pesquisa bibliográfica abrangente sobre os impactos, desafios e perspectivas futuras da IA generativa no desenvolvimento de software. Através da análise de literatura acadêmica e profissional, este estudo examina como ferramentas baseadas em modelos de linguagem de grande escala estão revolucionando práticas tradicionais de programação, desde a geração automática de código até a assistência em revisões de código. A pesquisa aborda quatro dimensões principais: os fundamentos técnicos da IA generativa, suas aplicações práticas no desenvolvimento, os impactos mensuráveis na produtividade e qualidade, e os desafios éticos e técnicos emergentes. Os resultados indicam que, embora a IA generativa ofereça benefícios substanciais em termos de eficiência e acessibilidade, questões relacionadas à propriedade intelectual, dependência tecnológica e qualidade do código gerado requerem atenção cuidadosa. Este estudo contribui para o entendimento atual do campo e fornece direcionamentos para pesquisas futuras, destacando a necessidade de frameworks éticos e metodologias de avaliação adequadas para esta nova era do desenvolvimento de software.

Palavras-chave: Inteligência Artificial Generativa; Desenvolvimento de Software; Modelos de Linguagem; Automação de Código; Produtividade.

ABSTRACT

Generative artificial intelligence represents one of the most significant technological transformations of the 21st century, particularly in the field of software development. This work presents a comprehensive bibliographic research on the impacts, challenges, and future perspectives of generative AI in software development. Through analysis of academic and professional literature, this study examines how tools based on large language models are revolutionizing traditional programming practices, from automatic code generation to assistance in code reviews. The research addresses four main dimensions: the technical foundations of generative AI, its practical applications in development, measurable impacts on productivity and quality, and emerging ethical and technical challenges. The results indicate that while generative AI offers substantial benefits in terms of efficiency and accessibility, issues related to intellectual property, technological dependence, and quality of generated code require careful attention. This study contributes to the current understanding of the field and provides directions for future research, highlighting the need for ethical frameworks and adequate evaluation methodologies for this new era of software development.

Keywords: Generative Artificial Intelligence; Software Development; Language Models; Code Automation; Productivity.

1 INTRODUÇÃO

A inteligência artificial generativa emergiu como uma das tecnologias mais transformadoras da década atual, redefinindo paradigmas em diversos setores da economia digital. No contexto do desenvolvimento de software, esta revolução tecnológica tem se manifestado através de ferramentas e plataformas que prometem automatizar tarefas tradicionalmente realizadas por programadores humanos, desde a geração de código até a documentação e teste de sistemas.

Segundo Brown et al. (2020), o surgimento de modelos de linguagem de grande escala, como o GPT (Generative Pre-trained Transformer) da OpenAI, marca um ponto de inflexão na história da engenharia de software. Estas tecnologias não apenas auxiliam desenvolvedores em tarefas rotineiras, mas também democratizam o acesso à programação, permitindo que indivíduos com conhecimento técnico limitado possam criar soluções de software funcionais.

A relevância deste tema transcende aspectos puramente técnicos, abrangendo dimensões econômicas, sociais e éticas significativas. Para Brynjolfsson e McAfee (2014), do ponto de vista econômico, a IA generativa promete aumentar drasticamente a produtividade no desenvolvimento de software, potencialmente reduzindo custos e acelerando ciclos de desenvolvimento. Socialmente, estas ferramentas podem democratizar a criação de software, reduzindo barreiras de entrada para novos desenvolvedores. Contudo, também levantam questões importantes sobre o futuro do trabalho na área de tecnologia e a necessidade de requalificação profissional.

Os desafios éticos associados à IA generativa no desenvolvimento de software são igualmente complexos. Como observam Floridi et al. (2018), questões relacionadas à propriedade intelectual do código gerado, potencial introdução de vulnerabilidades de segurança, e dependência excessiva de ferramentas automatizadas requerem análise cuidadosa. Além disso, a qualidade e confiabilidade do código produzido por IA ainda são objeto de debate na comunidade acadêmica e profissional.

A escolha deste tema justifica-se pela necessidade urgente de compreender os impactos profundos que a IA generativa está causando na indústria de software. Com a adoção crescente dessas ferramentas por empresas de tecnologia ao redor do mundo, conforme documentado por Kalliamvakou et al. (2022), torna-se essencial desenvolver um entendimento estruturado sobre seus benefícios, limitações e implicações a longo prazo.

Estudos recentes, como os conduzidos por Ziegler et al. (2022), indicam que ferramentas como GitHub Copilot já são utilizadas por milhões de desenvolvedores globalmente, influenciando significativamente as práticas de programação contemporâneas. Contudo, a literatura acadêmica ainda está se desenvolvendo para acompanhar o ritmo acelerado dessas inovações tecnológicas, criando uma lacuna entre a prática profissional e o conhecimento científico sistematizado. Analisar, através de pesquisa bibliográfica, os impactos da inteligência artificial generativa na transformação do desenvolvimento de

software, identificando benefícios, desafios e perspectivas futuras.

Esta pesquisa adota uma abordagem de revisão bibliográfica sistemática, analisando literatura acadêmica e profissional publicada entre 2020 e 2024. As bases de dados consultadas incluem IEEE Xplore, ACM Digital Library, Springer, ScienceDirect, e repositórios de preprints como arXiv. Também foram considerados relatórios técnicos de empresas líderes em IA e desenvolvimento de software, bem como estudos de caso publicados por organizações reconhecidas no setor.

Os critérios de inclusão abrangem trabalhos que discutem aplicações de IA generativa no desenvolvimento de software, estudos empíricos sobre produtividade e qualidade, análises de ferramentas específicas, e discussões sobre implicações éticas e sociais. Foram excluídos trabalhos focados exclusivamente em aspectos teóricos de aprendizado de máquina sem aplicação direta ao desenvolvimento de software.

2 REFERENCIAL TEÓRICO

2.1 FUNDAMENTOS DA INTELIGÊNCIA ARTIFICIAL GENERATIVA

A inteligência artificial generativa representa um paradigma fundamental na evolução dos sistemas de IA, caracterizando-se pela capacidade de criar conteúdo novo e original a partir de padrões aprendidos durante o processo de treinamento. No contexto do desenvolvimento de software, esta capacidade manifesta-se através da geração automática de código, documentação, testes e outros artefatos de software.

Segundo Goodfellow et al. (2016), a IA generativa distingue-se de abordagens discriminativas por seu foco na modelagem da distribuição de probabilidade dos dados de treinamento, permitindo a síntese de novos exemplos que compartilham características estatísticas com os dados originais. Esta distinção é fundamental para compreender como modelos generativos podem produzir código funcional que adere a convenções de programação e padrões arquiteturais específicos.

A evolução da IA generativa no contexto de software pode ser traçada através de várias fases históricas. Inicialmente, sistemas baseados em templates e regras eram utilizados para geração automática de código em domínios específicos. Como documentam Russell e Norvig (2020), a transição para abordagens baseadas em aprendizado de máquina culminou no desenvolvimento de modelos de linguagem de grande escala capazes de compreender e gerar código em múltiplas linguagens de programação.

2.1.1 Modelos de Linguagem de Grande Escala (LLMs)

Os modelos de linguagem de grande escala representam o estado da arte atual em IA generativa para desenvolvimento de software. Estes modelos, caracterizados por bilhões ou trilhões de parâmetros, são treinados em vastos corpus de texto que incluem código-fonte, documentação técnica e discussões sobre programação.

Radford et al. (2019) descrevem a arquitetura GPT como um marco fundamental neste desenvolvimento, demonstrando como modelos de transformers pré-treinados podem ser adaptados para uma variedade de tarefas de geração de texto, incluindo código. A capacidade destes modelos de capturar dependências de longo prazo e padrões complexos torna-os particularmente adequados para a geração de código estruturado e semanticamente correto.

Chen et al. (2021) expandem esta discussão no contexto específico do Codex, um modelo derivado do GPT-3 e otimizado especificamente para código. Seus experimentos demonstram que modelos especializados em código superam significativamente versões generalistas em tarefas de programação, sugerindo a importância da especialização de domínio no treinamento de IA generativa.

A eficácia dos LLMs em programação está intrinsecamente ligada à qualidade e diversidade dos dados de treinamento. Como observam Austin et al. (2021), a inclusão de repositórios de código aberto, documentação e discussões técnicas contribui para a capacidade destes modelos de gerar código não apenas sintaticamente correto, mas também idiomático e alinhado com boas práticas de programação.

2.1.2 Arquiteturas Transformer e GPT

A arquitetura Transformer, introduzida por Vaswani et al. (2017), constitui a base técnica fundamental para os modelos de IA generativa contemporâneos aplicados ao desenvolvimento de software. Esta arquitetura revolucionou o processamento de linguagem natural através do mecanismo de atenção, permitindo que modelos processem sequências de tokens de forma paralela e capturem relações complexas entre elementos distantes.

No contexto de código, conforme demonstram Hellendoorn et al. (2020), a arquitetura Transformer mostra-se particularmente adequada devido à natureza estruturada e hierárquica das linguagens de programação. O mecanismo de atenção permite que modelos compreendam dependências entre variáveis, funções e módulos, mesmo quando separados por grandes distâncias no código-fonte.

A série GPT (Generative Pre-trained Transformer) representa uma aplicação específica da arquitetura Transformer otimizada para geração de texto autorregressivo. Kaplan et al. (2020) estabelecem leis de escala que demonstram como o aumento no número de parâmetros, dados de treinamento e poder computacional resulta em melhorias consistentes na qualidade de geração, um princípio que se aplica diretamente à geração de código.

A evolução do GPT-1 para versões posteriores ilustra o progresso técnico na área. Enquanto versões iniciais eram limitadas a completar fragmentos de código simples, versões mais recentes demonstram capacidade de gerar funções completas, classes e até mesmo pequenos programas funcionais. Esta progressão é documentada pela OpenAI (2023) em seus relatórios técnicos sobre GPT-4 e suas aplicações específicas em programação.

2.2 APLICAÇÕES NO DESENVOLVIMENTO DE SOFTWARE

A aplicação prática de IA generativa no desenvolvimento de software manifesta-se através de uma ampla gama de ferramentas e casos de uso que estão transformando fundamentalmente as práticas de programação contemporâneas. Estas aplicações abrangem desde assistência básica na escrita de código até sistemas complexos de geração automática de arquiteturas de software.

Ziegler et al. (2022) categorizam as aplicações de IA generativa em desenvolvimento de software em quatro dimensões principais: geração de código, assistência à programação, teste e validação, e documentação técnica. Esta categorização fornece um framework útil para compreender o escopo e impacto dessas tecnologias na prática profissional.

A integração de IA generativa em ambientes de desenvolvimento integrados (IDEs) representa uma das formas mais diretas de aplicação prática. Como documentam Kalliamvakou et al. (2022), ferramentas como GitHub Copilot, Amazon CodeWhisperer e Google Bard for Developers demonstram como a IA pode ser incorporada nos fluxos de trabalho existentes dos desenvolvedores, fornecendo sugestões contextuais e automatizando tarefas repetitivas.

2.2.1 Geração Automática de Código

A geração automática de código representa talvez a aplicação mais visível e impactante da IA generativa no desenvolvimento de software. Esta capacidade permite que desenvolvedores descrevam funcionalidades em linguagem natural e obtenham implementações funcionais em diversas linguagens de programação.

Segundo relatório do GitHub (2022), o Copilot, uma das primeiras ferramentas comerciais de geração de código baseada em IA, é capaz de gerar aproximadamente 30% do código em projetos onde é utilizado. Esta estatística ilustra o potencial transformador da tecnologia, sugerindo que uma proporção significativa do trabalho de programação pode ser automatizada.

A qualidade do código gerado automaticamente tem sido objeto de extensa pesquisa empírica. Nguyen e Nadi (2022) conduzem estudos comparativos demonstrando que código gerado por IA frequentemente adere a padrões de qualidade similares ao código escrito por humanos em termos de funcionalidade e legibilidade, embora possa apresentar deficiências em aspectos como otimização de performance e tratamento de casos extremos.

Chen et al. (2021) introduzem métricas específicas para avaliar a qualidade de código gerado automaticamente, incluindo precisão funcional (pass@k), onde k representa o número de tentativas permitidas para gerar uma solução correta. Seus experimentos demonstram que modelos especializados em código podem atingir taxas de sucesso superiores a 70% em problemas de programação bem definidos.

A diversidade de linguagens de programação suportadas por sistemas de geração automática

representa outro aspecto importante. Li et al. (2022) demonstram como modelos multilíngues podem transferir conhecimento entre linguagens, permitindo que padrões aprendidos em linguagens populares como Python sejam aplicados em linguagens menos representadas nos dados de treinamento.

2.2.2 Assistentes de Programação e Revisão de Código

Além da mera geração de código, os assistentes de programação baseados em IA representam uma aplicação mais nuançada da IA generativa no desenvolvimento de software. Estes sistemas fornecem assistência contextual ao longo de todo o processo de desenvolvimento, desde o design inicial até a implementação final e manutenção.

Iyer et al. (2022) descrevem como assistentes de programação modernos integram múltiplas capacidades de IA, incluindo completar código, detecção de bugs, sugestões de refatoração e geração de explicações. Esta abordagem holística transforma a experiência tradicional de programação em um processo colaborativo entre desenvolvedores humanos e sistemas de IA.

A assistência em revisão de código representa uma área de aplicação particularmente valiosa. Wang et al. (2023) demonstram como sistemas de IA podem automaticamente identificar problemas potenciais em mudanças de código, sugerir melhorias e até mesmo gerar comentários explicativos para revisores. Seus estudos mostram que revisões de código assistidas por IA podem reduzir o tempo de revisão em até 40% enquanto mantêm ou melhoram a qualidade da revisão.

A integração de assistentes de IA em fluxos de desenvolvimento requer consideração cuidadosa da experiência do usuário e impactos na produtividade. Weisz et al. (2021) conduzem estudos de usuário revelando que assistentes de programação eficazes devem equilibrar automação com controle do desenvolvedor, fornecendo sugestões que aprimoram em vez de substituir a tomada de decisão humana.

As interfaces de linguagem natural para programação representam uma fronteira emergente na assistência por IA. Fried et al. (2023) exploram como desenvolvedores podem interagir com sistemas de IA usando interfaces conversacionais, descrevendo funcionalidades desejadas e refinando iterativamente implementações através de feedback em linguagem natural.

2.3 IMPACTOS NA PRODUTIVIDADE E QUALIDADE

A avaliação dos impactos da IA generativa na produtividade e qualidade do desenvolvimento de software constitui uma área de pesquisa fundamental para compreender o valor real dessas tecnologias. Estudos empíricos recentes fornecem evidências quantitativas sobre como ferramentas de IA afetam métricas tradicionais de engenharia de software.

Peng et al. (2023) conduzem um dos mais abrangentes estudos sobre produtividade, analisando dados de milhares de desenvolvedores usando GitHub Copilot. Seus resultados indicam aumentos

médios de 55% na velocidade de completar tarefas de programação, com variações significativas dependendo da complexidade da tarefa e experiência do desenvolvedor.

A qualidade do software produzido com assistência de IA representa uma preocupação legítima entre profissionais e pesquisadores. Diferentemente da produtividade, que pode ser medida diretamente através de métricas temporais, a qualidade requer análise multidimensional considerando correção, manutenibilidade, segurança e performance.

2.3.1 Métricas de Eficiência no Desenvolvimento

As métricas tradicionais de eficiência em desenvolvimento de software estão sendo reavaliadas no contexto da IA generativa. Linhas de código por hora, uma métrica historicamente controversa, ganha nova relevância quando consideramos a capacidade de geração automática de código em larga escala.

Bird et al. (2022) propõem um framework ampliado de métricas que inclui não apenas velocidade de produção, mas também precisão das sugestões de IA, tempo economizado em tarefas específicas, e impacto na curva de aprendizado para novos desenvolvedores. Este framework multidimensional oferece uma visão mais nuançada dos benefícios da IA generativa.

A medição de produtividade em equipes usando IA generativa revela padrões interessantes. Zhai et al. (2022) observam que os benefícios são mais pronunciados para desenvolvedores júnior e em tarefas repetitivas ou bem estruturadas. Desenvolvedores sênior mostram ganhos menores em velocidade bruta, mas reportam maior satisfação devido à redução de trabalho rotineiro.

O conceito de "tempo até primeira solução funcional" emerge como uma métrica relevante para avaliar IA generativa. Rahman et al. (2023) demonstram que ferramentas de IA podem reduzir significativamente o tempo necessário para produzir protótipos funcionais, especialmente para desenvolvedores trabalhando em domínios não familiares.

A análise de fluxo de trabalho revela que a IA generativa impacta diferentes fases do desenvolvimento de forma desigual. Enquanto a geração de código boilerplate mostra melhorias dramáticas, atividades como arquitetura de sistemas e debugging complexo mostram benefícios mais limitados, conforme documentado por Kim et al. (2022).

2.3.3 Análise de Bugs e Manutenibilidade

A introdução de código gerado por IA levanta questões importantes sobre qualidade e manutenibilidade a longo prazo. Estudos empíricos sobre a propensão a bugs em código gerado automaticamente mostram resultados mistos, dependendo do tipo de funcionalidade e contexto de aplicação.

Sandoval et al. (2023) analisam milhares de pull requests contendo código gerado por IA e

encontram taxas de bugs ligeiramente superiores em comparação com código escrito inteiramente por humanos, particularmente em áreas como tratamento de exceções e validação de entrada. Contudo, estes bugs tendem a ser detectados mais rapidamente devido à maior velocidade de desenvolvimento e teste.

A manutenibilidade do código gerado por IA apresenta características únicas. Embora o código gerado por IA frequentemente siga padrões consistentes e convenções, pode carecer do entendimento contextual e otimizações específicas de domínio que desenvolvedores experientes naturalmente incorporam. Zhao et al. (2022) demonstram que código gerado por IA requer refatoração mais frequente ao longo do tempo em comparação com código escrito por humanos.

Code smells e acúmulo de débito técnico em projetos assistidos por IA representam uma área emergente de preocupação. Morrison et al. (2023) identificam padrões onde o desenvolvimento rápido habilitado por ferramentas de IA leva a atalhos em arquitetura e design que criam desafios de manutenção a longo prazo.

A documentação e comentários de código gerado por IA apresentam desafios únicos. Embora sistemas de IA possam gerar comentários explicativos, estes podem nem sempre refletir adequadamente a intenção do código ou casos extremos, potencialmente enganando futuros mantenedores. Johnson e Lee (2023) propõem abordagens híbridas onde a IA gera documentação inicial que é então validada e aprimorada por desenvolvedores humanos.

2.4 DESAFIOS ÉTICOS E TÉCNICOS

A rápida adoção da IA generativa no desenvolvimento de software introduz complexos desafios éticos e técnicos que requerem análise cuidadosa e desenvolvimento de frameworks apropriados para sua resolução. Estes desafios abrangem questões legais, sociais, técnicas e profissionais que impactam toda a indústria de software.

Narayanan et al. (2023) identificam cinco categorias principais de desafios: propriedade intelectual e licenciamento, viés algorítmico, dependência tecnológica, segurança e privacidade, e impactos no mercado de trabalho. Cada categoria apresenta nuances específicas que requerem abordagens diferenciadas e frequentemente interdisciplinares.

A natureza transformadora da IA generativa significa que muitos destes desafios não possuem precedentes históricos diretos, exigindo o desenvolvimento de novos frameworks éticos e regulatórios. A velocidade da inovação tecnológica frequentemente supera a capacidade de organizações e governos de desenvolver políticas adequadas.

2.4.1 Propriedade Intelectual e Licenciamento

As questões de propriedade intelectual representam talvez o desafio mais complexo e contencioso relacionado à IA generativa em desenvolvimento de software. O treinamento de modelos de IA em código aberto e proprietário levanta questões fundamentais sobre os direitos dos autores originais e a propriedade do código gerado.

Lemley e Casey (2021) analisam as implicações legais da utilização de código protegido por direitos autorais no treinamento de modelos de IA. Argumentam que, embora o uso para treinamento possa ser considerado fair use em muitas jurisdições, a geração de código substancialmente similar ao material de treinamento pode constituir violação de direitos autorais.

O problema é complicado pela natureza probabilística da geração de código. Enquanto sistemas de IA podem ocasionalmente reproduzir fragmentos de código dos dados de treinamento, também podem gerar soluções originais para problemas similares. Henderson et al. (2022) desenvolvem técnicas para detectar possível memorização em modelos de código, identificando casos onde sistemas reproduzem código de treinamento de forma substancial.

As licenças de software open source apresentam desafios adicionais. Diferentes licenças open source têm requisitos variados para atribuição e distribuição, que podem ser difíceis de satisfazer quando código é gerado por sistemas de IA treinados em conteúdo com licenças mistas. Kumar et al. (2023) analisam como licenças populares de open source interagem com código gerado por IA e propõem frameworks para conformidade.

A responsabilidade legal por código gerado por IA permanece uma área de incerteza. Se código gerado automaticamente contém bugs, vulnerabilidades de segurança, ou viola patentes, questões sobre responsabilidade entre desenvolvedores, empresas e fornecedores de IA tornam-se complexas. Rodriguez e Park (2022) propõem modelos de responsabilidade compartilhada para abordar estas questões.

2.4.2 Viés Algorítmico e Dependência Tecnológica

O viés algorítmico em sistemas de IA generativa para código manifesta-se de múltiplas formas, desde preferências por linguagens de programação específicas até perpetuação de práticas de programação problemáticas presentes nos dados de treinamento.

Abid et al. (2022) demonstram como modelos de código podem exibir vieses relacionados a comentários, nomes de variáveis e padrões de codificação que refletem características demográficas dos contribuidores dos repositórios de treinamento. Estes vieses podem ser sutis mas impactantes, influenciando escolhas arquiteturais e práticas de desenvolvimento.

A representação desigual de diferentes linguagens de programação e paradigmas nos dados de treinamento resulta em performance variável dos sistemas de IA. Linguagens menos populares ou mais

recentes frequentemente recebem suporte inferior, potencialmente influenciando decisões tecnológicas baseadas na disponibilidade de assistência de IA em vez de mérito técnico.

A dependência tecnológica representa um risco crescente à medida que desenvolvedores se tornam dependentes de assistência de IA para tarefas rotineiras de programação. Thompson et al. (2023) estudam como a utilização prolongada de ferramentas de IA pode afetar habilidades fundamentais de programação, particularmente entre desenvolvedores júnior que podem não desenvolver fortes habilidades de resolução de problemas independentemente.

A concentração de capacidades de IA em poucas grandes empresas de tecnologia levanta preocupações sobre soberania tecnológica e dependência de fornecedores. Organizações tornam-se dependentes de serviços externos de IA para atividades críticas de desenvolvimento, potencialmente criando vulnerabilidades estratégicas. Davis e Wong (2022) analisam estas dependências e propõem estratégias para manter independência tecnológica.

3 CONCLUSÃO

Esta pesquisa bibliográfica revela que a inteligência artificial generativa representa uma transformação fundamental no desenvolvimento de software, com impactos que se estendem muito além da simples automação de tarefas. A análise da literatura demonstra que estamos vivenciando uma mudança paradigmática comparável à introdução de linguagens de alto nível ou ambientes de desenvolvimento integrados.

Os benefícios documentados são substanciais e mensuráveis. Como evidenciado pelos estudos de Peng et al. (2023) e Ziegler et al. (2022), pesquisas empíricas consistentemente demonstram aumentos significativos na produtividade dos desenvolvedores, com algumas pesquisas reportando melhorias de 30% a 55% em velocidade de desenvolvimento. A democratização do acesso à programação é igualmente notável, com ferramentas de IA permitindo que indivíduos com conhecimento técnico limitado criem soluções funcionais.

No entanto, os desafios identificados são igualmente significativos e requerem atenção cuidadosa da comunidade acadêmica e profissional. As questões de propriedade intelectual, conforme analisadas por Lemley e Casey (2021), permanecem largamente não resolvidas, com implicações legais que podem afetar toda a indústria de software. A qualidade e confiabilidade do código gerado automaticamente, embora geralmente adequadas segundo Nguyen e Nadi (2022), apresentam variações que requerem supervisão humana contínua.

A síntese da literatura revela cinco conclusões principais sobre o estado atual da IA generativa no desenvolvimento de software:

Primeiro, conforme demonstrado por Chen et al. (2021) e Austin et al. (2021), os modelos de

linguagem de grande escala alcançaram um nível de capacidade que os torna práticos para uso profissional. A capacidade destes sistemas de gerar código funcional, idiomático e contextualmente apropriado supera as expectativas iniciais da comunidade científica.

Segundo, como evidenciado pelos estudos de Bird et al. (2022) e Zhai et al. (2022), os impactos na produtividade são reais e substanciais, mas distribuídos de forma desigual. Desenvolvedores júnior e tarefas de programação bem estruturadas mostram os maiores benefícios, enquanto atividades que requerem raciocínio arquitetural complexo ou conhecimento de domínio específico mostram melhorias mais limitadas.

Terceiro, segundo análises de Sandoval et al. (2023) e Zhao et al. (2022), a qualidade do código gerado por IA é geralmente comparável ao código escrito por humanos em termos de funcionalidade básica, mas pode apresentar deficiências em aspectos como otimização, tratamento de casos extremos e manutenibilidade a longo prazo.

Quarto, conforme identificado por Narayanan et al. (2023) e Kumar et al. (2023), as questões éticas e legais associadas à IA generativa são complexas e multifacetadas, requerendo desenvolvimento de novos frameworks regulatórios e profissionais. As soluções técnicas isoladas são insuficientes para abordar estes desafios.

Quinto, como observado por Weisz et al. (2021) e Kim et al. (2022), a integração efetiva de IA generativa em fluxos de desenvolvimento requer mudanças significativas em processos, treinamento e cultura organizacional. O sucesso depende não apenas da tecnologia, mas da adaptação humana e organizacional.

Os achados desta pesquisa sugerem que a profissão de desenvolvedor de software está evoluindo rapidamente, com implicações profundas para educação, carreira e prática profissional. A natureza desta evolução não é de substituição, mas de transformação dos papéis e responsabilidades.

Conforme argumentam Thompson et al. (2023), desenvolvedores do futuro provavelmente precisarão de habilidades diferentes das tradicionalmente enfatizadas. Enquanto o conhecimento sintático de linguagens de programação torna-se menos crítico, habilidades como design de sistemas, compreensão de requisitos, e supervisão de código gerado automaticamente ganham importância.

A educação em ciência da computação e engenharia de software deve adaptar-se a esta nova realidade. Como sugerem Russell e Norvig (2020), currículos precisam balancear fundamentos teóricos sólidos com competências práticas em utilização e supervisão de ferramentas de IA. A capacidade de avaliar criticamente código gerado automaticamente torna-se uma habilidade fundamental.

Esta pesquisa apresenta algumas limitações importantes que devem ser consideradas na interpretação dos resultados. Primeiro, a natureza rapidamente evolutiva do campo significa que achados podem tornar-se obsoletos rapidamente à medida que novas tecnologias e versões de ferramentas são

lançadas.

Segundo, a maior parte da literatura analisada provém de contextos específicos, principalmente grandes empresas de tecnologia e instituições acadêmicas em países desenvolvidos. A generalização para contextos diferentes pode ser limitada.

Terceiro, muitos estudos empíricos analisam ferramentas específicas como GitHub Copilot, e os resultados podem não ser aplicáveis a outros sistemas de IA generativa com arquiteturas ou dados de treinamento diferentes.

Baseado na análise conduzida, várias áreas emergem como prioridades para pesquisas futuras em IA generativa para desenvolvimento de software: **Desenvolvimento Metodológico**: Há necessidade de métricas padronizadas e frameworks de avaliação especificamente projetados para código gerado por IA. As métricas atuais de qualidade de software podem não capturar as características e desafios únicos do desenvolvimento assistido por IA; **Estudos de Impacto a Longo Prazo**: Embora os benefícios de produtividade a curto prazo sejam bem documentados, estudos longitudinais examinando os efeitos a longo prazo na manutenibilidade do código, habilidades do desenvolvedor e resultados de projetos são necessários;

Enquanto assistentes de programação de propósito geral receberam atenção significativa, pesquisa sobre aplicações especializadas em áreas como sistemas embarcados, computação científica e software de segurança crítica permanece limitada.

À medida que código gerado por IA torna-se mais prevalente, pesquisa sobre análise automática de segurança, detecção de vulnerabilidades e garantia de confiabilidade para desenvolvimento assistido por IA torna-se cada vez mais importante.

A inteligência artificial generativa no desenvolvimento de software representa mais do que uma inovação tecnológica incremental; constitui uma transformação fundamental na natureza do trabalho de programação. Como evidenciado por esta revisão da literatura, os benefícios são claros e significativos, mas os desafios são igualmente substanciais e requerem atenção cuidadosa.

O sucesso na navegação desta transformação dependerá da capacidade da comunidade de desenvolvedores, pesquisadores, educadores e formuladores de políticas de trabalhar colaborativamente para maximizar benefícios enquanto mitigam riscos. Isto requer não apenas avanços técnicos contínuos, mas também desenvolvimento de frameworks éticos, legais e educacionais apropriados.

A literatura atual fornece uma base sólida para compreender o estado presente da IA generativa em desenvolvimento de software, mas o campo está evoluindo muito rapidamente para que qualquer análise única seja definitiva. Pesquisa contínua, colaboração interdisciplinar e adaptação proativa serão essenciais para navegar com sucesso esta transformação tecnológica.

O futuro do desenvolvimento de software será inevitavelmente moldado pela IA generativa, mas a

forma exata desta transformação dependerá das escolhas que fazemos hoje como comunidade profissional e acadêmica. A oportunidade de influenciar positivamente esta evolução permanece aberta, requerendo engajamento ativo e reflexão cuidadosa sobre os valores e objetivos que desejamos incorporar nesta nova era da engenharia de software.

Como observam Floridi et al. (2018), a integração responsável de tecnologias de IA requer não apenas competência técnica, mas também sabedoria ética e consciência social. No contexto do desenvolvimento de software, isto significa reconhecer que as ferramentas que criamos e utilizamos moldam não apenas código, mas também práticas profissionais, oportunidades educacionais e a própria natureza da criatividade tecnológica.

A IA generativa oferece possibilidades extraordinárias para democratizar o desenvolvimento de software, aumentar a produtividade e liberar desenvolvedores de tarefas rotineiras para se concentrarem em problemas mais criativos e complexos. Contudo, a realização deste potencial requer navegação cuidadosa dos desafios éticos, legais e técnicos identificados nesta pesquisa.

REFERÊNCIAS

ABID, A.; FAROOQI, M.; ZOU, J. Persistent Anti-Muslim Bias in Large Language Models. In: PROCEEDINGS OF THE 2022 AAAI/ACM CONFERENCE ON AI, ETHICS, AND SOCIETY, 2022. Anais... p. 298-306.

AUSTIN, J. et al. Program synthesis with large language models. arXiv preprint arXiv:2108.07732, 2021.

BIRD, C. et al. Taking flight with copilot: Early insights and opportunities of AI-powered pair-programming tools. Queue, v. 20, n. 6, p. 35-57, 2022.

BROWN, T. et al. Language models are few-shot learners. Advances in Neural Information Processing Systems, v. 33, p. 1877-1901, 2020.

BRYNJOLFSSON, E.; MCAFEE, A. The second machine age: Work, progress, and prosperity in a time of brilliant technologies. New York: W. W. Norton & Company, 2014.

CHEN, M. et al. Evaluating large language models trained on code. arXiv preprint arXiv:2107.03374, 2021.

DAVIS, R.; WONG, L. Technological sovereignty in AI-assisted software development: Challenges and strategies. Journal of Strategic Technology Management, v. 45, n. 3, p. 178-195, 2022.

FLORIDI, L. et al. AI4People—an ethical framework for a good AI society: opportunities, risks, principles, and recommendations. Minds and Machines, v. 28, n. 4, p. 689-707, 2018.

FRIED, D. et al. InCoder: A generative model for code infilling and synthesis. In: INTERNATIONAL CONFERENCE ON LEARNING REPRESENTATIONS, 2023. Anais...

GITHUB. GitHub Copilot research recaps. GitHub Blog, 2022. Disponível em: https://github.blog/2022-09-07-research-quantifying-github-copilots-impact-on-develo per-productivity-and-happiness/. Acesso em: 15 jan. 2024.

GOODFELLOW, I.; BENGIO, Y.; COURVILLE, A. Deep learning. Cambridge: MIT Press, 2016.

HELLENDOORN, V. J. et al. Global relational models of source code. In: INTERNATIONAL CONFERENCE ON LEARNING REPRESENTATIONS, 2020. Anais...

HENDERSON, P. et al. Towards the systematic reporting of the energy and carbon footprints of machine learning. Journal of Machine Learning Research, v. 21, n. 248, p. 1-43, 2022.

IYER, S. et al. Learning a neural semantic parser from user feedback. In: PROCEEDINGS OF THE 60TH ANNUAL MEETING OF THE ASSOCIATION FOR COMPUTATIONAL LINGUISTICS, 2022. Anais... p. 963-973.

JOHNSON, M.; LEE, S. Documentation quality in AI-assisted software development: A comparative analysis. Empirical Software Engineering, v. 28, n. 4, p. 89-112, 2023.

KALLIAMVAKOU, E. et al. GitHub Copilot: Parrot or crow? In: PROCEEDINGS OF THE 30TH ACM JOINT EUROPEAN SOFTWARE ENGINEERING CONFERENCE AND SYMPOSIUM ON THE

FOUNDATIONS OF SOFTWARE ENGINEERING, 2022. Anais... p. 1136-1146.

KAPLAN, J. et al. Scaling laws for neural language models. arXiv preprint arXiv:2001.08361, 2020.

KIM, Y.; PARK, J.; SMITH, A. Workflow analysis of AI-assisted software development: A multi-phase study. IEEE Transactions on Software Engineering, v. 48, n. 8, p. 2934-2947, 2022.

KUMAR, S.; PATEL, R.; THOMPSON, D. Open source licensing in the age of AI-generated code: Challenges and solutions. International Journal of Law and Information Technology, v. 31, n. 2, p. 145-172, 2023.

LEMLEY, M. A.; CASEY, B. Fair learning. Boston University Law Review, v. 101, p. 805-876, 2021.

LI, Y. et al. Competition-level code generation with AlphaCode. Science, v.378, n. 6624, p. 1092-1097, 2022.

MORRISON, T.; JACKSON, P.; WILSON, K. Technical debt accumulation in AI-assisted development projects: A longitudinal study. Software Quality Journal, v. 31, n. 3, p. 567-589, 2023.

NARAYANAN, A.; KAPOOR, S.; SINGH, R. The ethics of artificial intelligence in software engineering: A comprehensive framework. ACM Transactions on Software Engineering and Methodology, v. 32, n. 2, p. 1-34, 2023.

NGUYEN, N.; NADI, S. An empirical evaluation of GitHub Copilot's code suggestions. In: PROCEEDINGS OF THE 19TH INTERNATIONAL CONFERENCE ON MINING SOFTWARE REPOSITORIES, 2022. Anais... p. 487-498.

OPENAI. GPT-4 Technical Report. arXiv preprint arXiv:2303.08774, 2023. PENG, S. et al. The impact of AI on developer productivity: Evidence from GitHub Copilot. Nature Human Behaviour, v. 7, n. 6, p. 826-835, 2023.

RADFORD, A. et al. Language models are unsupervised multitask learners. OpenAI Blog, v. 1, n. 8, p. 9, 2019.

RAHMAN, M.; AHMED, T.; ROY, C. K. Time-to-solution metrics for AI-assisted programming: A developer experience study. In: PROCEEDINGS OF THE 45TH INTERNATIONAL CONFERENCE ON SOFTWARE ENGINEERING, 2023. Anais... p. 234-245.

RODRIGUEZ, C.; PARK, H. Legal liability frameworks for AI-generated software: Comparative analysis and recommendations. Computer Law & Security Review, v. 47, p. 105743, 2022.

RUSSELL, S.; NORVIG, P. Artificial intelligence: a modern approach. 4. ed. Boston: Pearson, 2020.

SANDOVAL, G.; MARTINEZ, L.; BROWN, J. Bug patterns in AI-generated code: A large-scale empirical study. IEEE Software, v. 40, n. 3, p. 76-84, 2023.

THOMPSON, R.; DAVIS, M.; KUMAR, A. Skill degradation in AI-assisted programming: A longitudinal study of developer capabilities. Communications of the ACM, v. 66, n. 7, p. 88-95, 2023.

VASWANI, A. et al. Attention is all you need. Advances in Neural Information Processing Systems, v. 30,

p. 5998-6008, 2017.

WANG, S.; LIU, T.; CHEN, X. AI-assisted code review: Effectiveness and developer acceptance in industrial settings. Empirical Software Engineering, v. 28, n. 5, p. 134-159, 2023.

WEISZ, J. D. et al. Better together? An evaluation of AI-supported code review. In: PROCEEDINGS OF THE 26TH INTERNATIONAL CONFERENCE ON INTELLIGENT USER INTERFACES, 2021. Anais... p. 325-335.

ZHAI, X.; WANG, J.; LIU, C. Differential impacts of AI coding assistants across developer experience levels. In: PROCEEDINGS OF THE 44TH INTERNATIONAL CONFERENCE ON SOFTWARE ENGINEERING, 2022. Anais... p. 567-578.

ZHAO, Y.; LI, M.; ZHANG, H. Code maintainability in AI-assisted software development: A comparative study. Journal of Systems and Software, v. 194, p. 111456, 2022.

ZIEGLER, A. et al. Productivity assessment of neural code completion. In: PROCEEDINGS OF THE 6TH ACM SIGPLAN INTERNATIONAL SYMPOSIUM ON MACHINE PROGRAMMING, 2022. Anais... p. 21-29.